AUSTRALIAN OS9 NEWSLETTER

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-1284
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
SUPPORT	Brisbane OS9 Users Group	

Addresses for Correspondence

Editorial Material:

Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Subscriptions & Library Requests:

Jean-Pierre Jacquet
27 Hampton Street
DURACK Qld 4077

Volume 5 July 1991 Number 6

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 5 Number 6

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet
SUPPORT : Brisbane OS9 Level 2 Users Group.

This edition of our newsletter brings us close to the end of the third year of production from sunny Queensland. Our next (August) edition will complete the three years, and it is now time to start thinking of the new subscription year.

Members wishing to re-subscribe will need to do so before the end of August. We will maintain the same subscription rate of $18.00 for the coming year. That's no increase for four years!! How is that? This has been made possible by the number of $2 copy fees received for public domain material.

Current membership is 63 and we suspect that some of you will not wish to renew as you have moved to some other computer type and operating system. However, we will require a minimum of 20 members to make it worthwhile continuing.

Some of you have joined the OS9 Usergroup only recently and we should have supplied you with back issues from September 1990. I have slipped up more than once in this regard, so if we still owe you any back issues, please drop me a line and I will correct the situation. Any new subscribers from now on will be treated as members for the year commencing September 1991 and will be added to the mailing list immediately. This may mean one or two free issues, but it is just too difficult to maintain an endless supply of back issues. Back issues will now only be available as disk files.

A membership application/renewal form is included with this newsletter.

PUBLIC DOMAIN LIBRARY

Just a reminder that we have a database of the P.D. library, UGCAT (UserGroup Catalogue) which is available on CoCo OS9 format floppy and Atari 3.5 format.

Both versions of UGCAT are complete with the Basic09 (CoCo OS9) or Basic (OSK) "Lookup" programme by Bob Devries. Please send requests to our Librarian together with formatted disk, return postage and $2 copy fee.

The P.D. library as it stands is in "ar" archives and comprises eleven 80 track D.S. (720k) disks.

IN THIS EDITION

We also have an update to the Newsletter CONTENTS listing. This is very handy if you are looking for a previous article.

We are most grateful to those members who have contributed articles for publication in these pages and encourage all members to "have a go" regardless of your experience level. Some newer Usergroup members are just starting the OS9 experience and will need some help. I am sure that many of us could tell a long story about our early frustrations and experiences, so how about a few tips for those new users.

I have received a few queries from members in trouble with some applications which have not yet been answered. Please be patient, we will get there.

Cheers, Gordon.

oooooooooo0000000000oooooooooo

A Basic09 Tutorial
by Bob Devries

Here is the BASIC code for the numbersquare programme from 'Microcomputing', June 1981. It is written in vanilla BASIC, but is suitable for Extended Colour Basic with minor modifications, in lines 140,490, 760, 920, 960 and 1450. Can you work out what to change? (Note please that a colon is used instead of a REM)

```
0010 : Number square game
0020 : ver 4.0 - 12 nov 79
0030 : Marc I. Leavey, M.D.
0040 LINE= 0
0050 DIGITS= 0
0060 PRINT "N U M B E R   S Q U A R E S"
0070 PRINT "---------------------------"
0080 PRINT
0090 PRINT "WELCOME TO THE WORLD OF"
0100 PRINT "CONFUSION.  THERE ARE TWO"
0110 PRINT "VERSIONS OF NUMBER SQUARES:"
0120 PRINT " 1 - SEQUENTIAL"
0130 PRINT " 2 - MAGIC SQUARE"
0140 INPUT "WHICH IS YOUR PLEASURE",T
0150 IF T=1 GOTO 310
0160 IF T<>2 GOTO 140
0170 :
0180 : SET UP MAGIC
0190 : SQUARE BOARD
0200 :
0210 FOR I=1 TO 4
0220 FOR J=1 TO 4
0230 READ M(I,J)
0240 LET B(I,J)=M(I,J)
0250 NEXT J
0260 NEXT I
0270 DATA 1,6,15,8,12,11,2,5,10,13,4,3,7,16,9,14
0280 LET I1=4
0290 LET J1=2
0300 GOTO 440
0310 :
0320 : SET UP SEQUENTIAL
0330 : BOARD
0340 :
0350 DIM B(4,4)
0360 FOR I=1 TO 4
0370 FOR J=1 TO 4
0380 LET B(I,J)=(I-1)*4+J
0390 NEXT J
0400 NEXT I
0410 LET I1=4
0420 LET J1=4
0430 :
0440 : NOW SCRAMBLE THE BOARD
0450 : TWO HUNDRED TIMES
0460 :
0470 PRINT "I AM NOW SCRAMBLING THE BOARD..."
0480 FOR Q=1 TO 200
0490 LET M=INT(1+RND*4)
0500 ON M GOTO 510,560,610,660
0510 IF I1=1 GOTO 490
0520 LET B(I1,J1)=B(I1-1,J1)
0530 LET B(I1-1,J1)=16
0540 LET I1=I1-1
0550 GOTO 700
0560 IF I1=4 GOTO 490
0570 LET B(I1,J1)=B(I1+1,J1)
0580 LET B(I1+1,J1)=16
0590 LET I1=I1+1
0600 GOTO 700
0610 IF J1=1 GOTO 490
0620 LET B(I1,J1)=B(I1,J1-1)
0630 LET B(I1,J1-1)=16
0640 LET J1=J1-1
0650 GOTO 700
0660 IF J1=4 GOTO 490
0670 LET B(I1,J1)=B(I1,J1+1)
0680 LET B(I1,J1+1)=16
0690 LET J1=J1+1
0700 NEXT Q
0710 :
0720 : PRINT BOARD
0730 :
0740 LET M9=0
0750 : OUTPUT A "HOME UP"
0760 PRINT CHR$(16);
0770 PRINT "---------------------"
0780 FOR I=1 TO 4
0790 FOR J=1 TO 4
0800 PRINT": ";
0810 IF B(I,J)=16 PRINT "   ";:GOTO 840
0820 IF B(I,J)<10 PRINT " ";
0830 PRINT B(I,J);
0840 NEXT J
0850 PRINT ":"
0860 PRINT "---------------------"
0870 NEXT I
0880 :
0890 : ERASE REST OF SCREEN AND
0900 : BEEP FOR INPUT
0910 :
0920 PRINT CHR$(22);CHR$(7);CHR$(7);
0930 :
0940 : INPUT MOVE
0950 :
0960 INPUT "MOVE WHICH PIECE",M
0970 LET I1=0:J1=0
0980 FOR I=1 TO 4
0990 FOR J=1 TO 4
1000 IF B(I,J)=M THEN I1=I:J1=J
```

```
1010 NEXT J
1020 NEXT I
1030 IF I1=0  THEN  PRINT  "I  CAN'T  FIND  THAT
NUMBER":GOTO 940
1040 LET I2=0:J2=0
1050 FOR I=I1-1 TO I1+1
1060 IF I>4 GOTO 1090
1070 IF I<1 GOTO 1090
1080 IF B(I,J1)=16 THEN I2=I:J2=J1:GOTO 1170
1090 NEXT I
1100 FOR J=J1-1 TO J1+1
1110 IF J>4 GOTO 1140
1120 IF J<1 GOTO 1140
1130 IF B(I1,J)=16 THEN I2=I1:J2=J:GOTO 1170
1140 NEXT J
1150 LET M9=M9+1
1160 PRINT "NOT A VALID MOVE":GOTO 940
1170 LET B(I2,J2)=K
1180 LET B(I1,J1)=16
1190 ON T GOTO 1210,1320
1200 :
1210 : SEQUENTIAL SOLUTION
1220 :
1230 LET C=0
1240 FOR I=1 TO 4
1250 FOR J=1 TO 4
1260 IF B(I,J)<C GOTO 720
1270 LET C=B(I,J)
1280 NEXT J
1290 NEXT I
1300 PRINT "YOU GOT IT!"
1310 GOTO 1450
1320 :
1330 : MAGIC SQUARE SOLUTION
1340 : CHECK
1350 :
1360 FOR I=1 TO 4
1370 FOR J=1 TO 4
1380 IF B(I,J)<>M(I,J) GOTO 720
1390 NEXT J
1400 NEXT I
1410 :
1420 : A WIN IS DECLARED!
1430 :
1440 PRINT "THAT IS THE CORRECT SOLUTION!"
1450 INPUT "LIKE TO PLAY ANOTHER GAME",I$
1460 IF LEFT$(I$,1)="Y" THEN RUN
1470 END
```

The game is a fairly simple one based on the use of multi-dimensioned arrays. Note the use of colons at the beginning of REM lines, which is also possible in DECB. Now comes the tricky part, the conversion to Basic09. Firstly I'll show you the code as I rewrote it, then I will explain it.

```
PROCEDURE numbersquare
BASE 1
(* version 4.0 - 12 NOV 79
(* Marc I. Leavey, MD
(* Basic09 version By Bob Devries April 91
DIM t:INTEGER
DIM i,j:INTEGER
DIM b(4,4):INTEGER
DIM i1,j1:INTEGER
DIM q,m,m9,c:INTEGER
DIM mm(4,4):INTEGER
DIM lop:BOOLEAN
DIM valid:BOOLEAN
DIM solution:BOOLEAN
DIM newgame:BOOLEAN
SHELL "tmode -pause"
newgame=TRUE
WHILE newgame=TRUE DO
PRINT CHR$(12);
PRINT "N U M B E R   S Q U A R E S"
PRINT "---------------------------"
PRINT
PRINT "Welcome to the world of"
PRINT "confusion.  There are two"
PRINT "versions of number squares:"
PRINT " 1 - sequential"
PRINT " 2 - Magic Square"
INPUT "Which is your pleasure ? ",t
IF t=1 THEN
RUN setupssb(b,i1,j1)
ELSE
RUN setupmsb(b,mm,i1,j1)
ENDIF
PRINT "I am now scrambling the board..."
FOR q=1 TO 200
lop=FALSE
WHILE lop=FALSE DO
m=1+RND(3)
IF m=1 THEN
IF i1<>1 THEN
b(i1,j1)=b(i1-1,j1)
b(i1-1,j1)=16
i1=i1-1
lop=TRUE
ENDIF
ENDIF
IF m=2 THEN
IF i1<>4 THEN
b(i1,j1)=b(i1+1,j1)
b(i1+1,j1)=16
i1=i1+1
lop=TRUE
ENDIF
ENDIF
IF m=3 THEN
IF j1<>1 THEN
b(i1,j1)=b(i1,j1-1)
```

```
b(i1,j1-1)=16
j1=j1-1
lop=TRUE
ENDIF
ENDIF
IF m=4 THEN
IF j1<>4 THEN
b(i1,j1)=b(i1,j1+1)
b(i1,j1+1)=16
j1=j1+1
lop=TRUE
ENDIF
ENDIF
ENDWHILE
NEXT q
(* print board line 720
solution=FALSE
REPEAT
m9=0
RUN gfx2("cwarea",29,12,22,12)
PRINT "-----------------"
FOR i=1 TO 4
FOR j=1 TO 4
PRINT ": ";
IF b(i,j)=16 THEN
PRINT "  ";
ELSE
IF b(i,j)<10 THEN
PRINT " ";
ENDIF
PRINT b(i,j);
ENDIF
NEXT j
PRINT ":"
PRINT "-----------------"
NEXT i
(* input move
valid=FALSE
WHILE valid=FALSE DO
i1=0
j1=0
WHILE i1=0 DO
RUN gfx2("bell")
INPUT "Move which piece ? ",m
IF m=0 THEN
RUN gfx2("cwarea",0,0,80,24)
PRINT CHR$(12)
SHELL "tmode pause"
END
ENDIF
FOR i=1 TO 4
FOR j=1 TO 4
IF b(i,j)=m THEN
i1=i
j1=j
ENDIF
NEXT j
NEXT i
IF i1=0 THEN
PRINT "I can't find that number"
ENDIF
ENDWHILE
i2=0
j2=0
FOR i=i1-1 TO i1+1
IF i>=1 AND i<=4 THEN
EXITIF b(i,j1)=16 THEN
i2=i
j2=j1
valid=TRUE
ENDEXIT
ENDIF
NEXT i
IF valid=FALSE THEN
FOR j=j1-1 TO j1+1
IF j>=1 AND j<=4 THEN
EXITIF b(i1,j)=16 THEN
i2=i1
j2=j
valid=TRUE
ENDEXIT
ENDIF
NEXT j
ENDIF
IF valid=FALSE THEN
m9=m9+1
PRINT "Not a valid move"
ENDIF
ENDWHILE
b(i2,j2)=m
b(i1,j1)=16
IF t=1 THEN
c=0
FOR i=1 TO 4
FOR j=1 TO 4
IF b(i,j)<c THEN (* reprint board
solution=FALSE
ENDIF
c=b(i,j)
NEXT j
NEXT i
IF solution=TRUE THEN
PRINT "You got it!"
ENDIF
ENDIF
IF t=2 THEN
FOR i=1 TO 4
FOR j=1 TO 4
IF b(i,j)<>mm(i,j) THEN (* reprint board
solution=FALSE
ENDIF
NEXT j
NEXT i
IF solution=TRUE THEN
```

```
PRINT "That is the correct solution!"
ENDIF
ENDIF
UNTIL solution=TRUE
INPUT "Like to play another game ? ",i$
IF LEFT$(i$,1)="n" THEN (* rerun game
newgame=FALSE
ENDIF
ENDWHILE
RUN gfx2("cwarea",0,0,80,24)
SHELL "tmode pause"
END

PROCEDURE setupssb
BASE 1
PARAM b(4,4):INTEGER
PARAM i1,j1:INTEGER
DIM i,j:INTEGER
FOR i=1 TO 4
FOR j=1 TO 4
b(i,j)=(i-1)*4+j
NEXT j
NEXT i
i1=4
j1=4

PROCEDURE setupmsb
BASE 1
PARAM b(4,4),mm(4,4):INTEGER
PARAM i1,j1:INTEGER
DIM i,j:INTEGER
FOR i=1 TO 4
FOR j=1 TO 4
READ mm(i,j)
b(i,j)=mm(i,j)
NEXT j
NEXT i
i1=4
j1=2
DATA 1,6,15,8,12,11,2,5,10,13,4,3,7,16,9,14
```

OK, so here we go. First, I used the command BASE 1. This is because all the arrays in the BASIC programme start at 1, and Basic09 usually starts at zero (actually, BASIC does too, but I see no reason to waste valuable memory). Next, I dimensioned all the variables and arrays, including a variable type which you may not have seen before, the BOOLEAN type. This variable may only contain either TRUE or FALSE! I used the SHELL command to turn off the OS9 screen pause, so that the programme won't sit there waiting at what it thinks is the end of a screen. That can get a bit confusing!

The next thing you must realise is that I have used NO LINE NUMBERS! This is really the best way to programme. Sure, Basic09 will allow their use, but the code is much more elegant without them, if a little more difficult to write. I set up a loop to allow the choice of whether to play another game which is done in line 1450 in the BASIC version. I used a WHILE loop here so that all I need to do is make a variable FALSE to exit the loop.

Next, after clearing the screen (printing a formfeed character), I print the opening message and ask the player for his choice of game. This follows through to line 160 of the BASIC programme. On the basis of the player's answer I RUN either the procedure 'setupssb' or 'setupmsb'. You may notice a slight variation here, I only tested for a '1' to select sequential, and any other key would run the magic square option. Then the next 38 lines do the same as lines 480 to 700 in the BASIC programme. You will notice that the BASIC programme uses quite a large number of GOTOs in this piece of code to break out of various parts of the code to continue the FOR-NEXT loop. I simply set a variable (lop) to TRUE and again used a WHILE loop. All the other variables have the same names, although you can of course use any name, and are not limited to two significant characters as in BASIC.

To make the screen easier to display, I used in this case a gfx2 command 'cwarea'. This command limits the area of the screen to be printed to, and means I don't need to use cursor manoeuvering code at every print line. I simply re-sized the screen to a 22 by 12 character box in approximately the middle of the (80 by 24) screen. So next I print the scrambled array to the screen with one space so that pieces may be moved around. The game is played by entering the number of the square which you want to move into the space. The programme checks to see if the number is one of the ones adjoining the space. A tone is sounded, and you are prompted for input. Pretty standard here, 'though I could just have used 'PRINT CHR$(7)', but the gfx2 'bell' command is nicer. If the player enters zero, the programme resets the screen back to its 80 by 24 size, and clears the screen and quits.

The rest of the code is fairly straight forward, again the original uses many GOTOs to quit out of FOR-NEXT loops (not a good practice in my opinion, even in BASIC), and I have used boolean tests for this purpose. For every move made, the programme checks the array against the solution, and if the last move solves the puzzle it prints the necessary result.

One of the hardest parts of the conversion is keeping track of the variables, and the various loops. Basic09 is a bit unforgiving about 'UNMATCHED CONTROL STRUCTURES' so you can't stop doing a conversion such as this in the middle, without generating a series of error messages when you quit the editor. One way around this is to use a text editor (like VED or SCRED or SLED) to create the source code first, and then to load it into Basic09. The only thing you MUST do in this case is to make the word 'PROCEDURE' in UPPERCASE the FIRST WORD in the file. The letter P of procedure must be the first character in the file, or Basic09 will not recognize it.

OK, so there you have it. I would love to hear from you regarding your own trials and tribulations with Basic09 programmes, even if you don't really want to start out on conversions of this type, but are having difficulties managing some aspect of Basic09. Please write to me care of the newsletter editor, and let 'Professor Bob' help sharpen your programming skills.

Regards,
Bob Devries

ooooooooooOOOOOOOOOOoooooooooo

Chapter 2 - Getting started in C
YOUR FIRST C PROGRAM

[ED: The text for this tutorial mentions some C programmes. These are not reproduced here, but are available on our Public Domain disk number 11]

The best way to get started with C is to actually look at a program, so load the file named TRIVIAL.C into your editor and display it on the monitor. You are looking at the simplest possible C program. There is no way to simplify this program or to leave anything out. Unfortunately, the program doesn't do anything. The word "main" is very important, and must appear once, and only once in every C program. This is the point where execution is begun when the program is run. We will see later that this does not have to be the first statement in the program but it must exist as the entry point.

Following the "main" program name is a pair of parentheses which are an indication to the compiler that this is a function. We will cover exactly what a function is in due time. For now, I suggest that you simply include the pair of parentheses. The two curly brackets, properly called braces, are used to define the limits of the program itself. The actual program statements go between the two braces and in this case, there are no statements because the program does absolutely nothing. You can compile and run this program, but since it has no executable statements, it does nothing. Keep in mind however, that it is a valid C program.

A PROGRAM THAT DOES SOMETHING

For a much more interesting program, load the program named WRTSOME.C and display it on your monitor. It is the same as the previous program except that it has one executable statement between the braces. The executable statement is another function. Once again, we will not worry about what a function is, but only how to use this one. In order to output text to the monitor, it is put within the function parentheses and bounded by quotation marks. The end result is that whatever is included between the quotation marks will be displayed on the monitor when the program is run. Notice the semi-colon at the end of the line. C uses a semi-colon as a statement terminator, so the semi-colon is required as a signal to the compiler that this line is complete. This program is also executable, so you can compile and run it to see if it does what you think it should.

ANOTHER PROGRAM WITH MORE OUTPUT

Load the program WRTMORE.C and display it on your monitor for an example of more output and another small but important concept. You will see that there are four program statements in this program, each one being a "printf" function statement. The top line will be executed first, then the next, and so on, until the fourth line is complete. The statements are executed in order from top to bottom. Notice the funny character near the end of the first line, namely the backslash. The backslash is used in the printf statement to indicate a special control character is following. In this case, the "n" [illegible] that [illegible] is requested. This is an indication to return the cursor to the left side of the monitor and move down one line. It is commonly referred to as a carriage return/line feed. Any place within text that you desire, you can put a newline character and start a new line. You could even put it in the middle of a word and split the word between two lines. The C compiler considers the combination of the backslash and letter n as one character.

A complete description of this program is now possible. The first printf outputs a line of text and returns the carriage. The second printf outputs a line but does not return the carriage so the third line is appended to that of the second, then followed by two carriage returns, resulting in a blank line. Finally the fourth printf outputs a line followed by a carriage return and the program is complete. Compile and run this program to see if it does what you expect it to do. It would be a good idea at this time for you to experiment by adding additional lines of printout to see if you understand how the statements really work.

LETS PRINT SOME NUMBERS

Load the file named ONEINT.C and display it on the monitor for our first example of how to work with data in a C program. The entry point "main" should be clear to you by now as well as the beginning brace. The first new thing we encounter is the line containing "int index;", which is used to define an integer variable named "index". The "int" is a reserved word in C, and can therefore not be used for anything else. It defines a variable that can have a value from -32768 to 32767 on most microcomputer implementations of C. Consult your users manual for the exact definition for your compiler. The variable name, "index", can be any name that follows the rules for an identifier and is not one of the reserved words for C. Consult your manual for an exact definition of an identifier for your compiler. The final character on the line, the semi-colon, is the statement terminator used in C.

We will see in a later chapter that additional integers could also be defined on the same line, but we will not complicate the present situation. Observing the main body of the program, you will notice that there are three statements that assign a value to the variable "index", but only one at a time. The first one assigns the value of 13 to "index", and its value is printed out. (We will see how shortly.) Later, the value of 27 is assigned to "index", and finally 10 is assigned to it, each value being printed out. It should be intuitively clear that "index" is indeed a variable and can store many different values. Please note that many times the words "printed out" are used to mean "displayed on the monitor". You will find that in many cases experienced programmers take this liberty, probably due to the "printf" function being used for monitor display.

HOW DO WE PRINT NUMBERS

To keep our promise, let's return to the "printf" statements for a definition of how they work. Notice that they are all identical and that they all begin just like the "printf" statements we have seen before. The first difference occurs when we come to the % character. This is a special character that signals the output routine to stop copying characters to the output and do something different, namely output a variable. The % sign is used to signal the start of many different types of variables, but we will restrict ourselves to only one for this example. The character following the % sign is a "d", which signals the output routine to get a decimal value and output it. Where the decimal value comes from will be covered shortly. After the "d", we find the familiar \n, which is a signal to return the video "carriage", and the closing quotation mark.

All of the characters between the quotation marks define the pattern of data to be output by this statement, and after the pattern, there is a comma followed by the variable name "index". This is where the "printf" statement gets the decimal value which it will output because of the "%d" we saw earlier. We could add more "%d" output field descriptors within the brackets and more variables following the description to cause more data to be printed with one statement. Keep in mind however, that it is important that the number of field descriptors and the number of variable definitions must be the same or the runtime system will get confused and probably quit with a runtime error. Much more will be covered at a later time on all aspects of input and output formatting. A reasonably good grasp of this topic is necessary in order to understand the following lessons. It is not necessary to understand everything about output formatting at this time, only a fair understanding of the basics. Compile and run ONEINT.C and observe the output.

HOW DO WE ADD COMMENTS IN C

Load the file COMMENTS.C and observe it on your monitor for an example of how comments can be added to a C program. Comments are added to make a program more readable to you but the compiler must ignore the comments. The slash star combination is used in C for comment delimiters. They are illustrated in the program at hand. Please note that the program does not illustrate good commenting practice, but is intended to illustrate where comments can go in a program. It is a very sloppy looking program. The first slash star combination introduces the first comment and the star slash at the end of the first line terminates this comment. Note that this comment is prior to the beginning of the program illustrating that a comment can precede the program itself. Good programming practice would include a comment prior to the program with a short introductory description of the program. The next comment is after the "main()" program entry point and prior to the opening brace for the program code itself. The third comment starts after the first executable statement and continues for four lines. This is perfectly legal because a comment can continue for as many lines as desired until it is terminated.

Note carefully that if anything were included in the blank spaces to the left of the three continuation lines of the comment, it would be part of the comment and would not be compiled. The last comment is located following the completion of the program, illustrating that comments can go nearly anywhere in a C program. Experiment with this program by adding comments in other places to see what will happen. Comment out one of the printf statements by putting comment delimiters both before and after it and see that it does not get printed out. Comments are very important in any programming language because you will soon forget what you did and why you did it. It will be much easier to modify or fix a well commented program a year from now than one with few or no comments. You will very quickly develop your own personal style of commenting. Some compilers allow you to "nest" comments which can be very handy if you need to "comment out" a section of code during debugging. Check your compiler documentation for the availability of this feature with you particular compiler. Compile and run COMMENTS.C at this time.

GOOD FORMATTING STYLE

Load the file GOODFORM.C and observe it on your monitor. It is an example of a well formatted program. Even though it is very short and therefore does very little, it is very easy to see at a glance what it does. With the experience you have already gained in this tutorial, you should be able to very quickly grasp the meaning of the program in it's entirety. Your C compiler ignores all extra spaces and all carriage returns giving you considerable freedom concerning how you format your program. Indenting and adding spaces is entirely up to you and is a matter of personal taste. Compile and run the program to see if it does what you expect it to do.

Now load and display the program UGLYFORM.C and observe it. How long will it take you to figure out what this program will do? It doesn't matter to the compiler which format style you use, but it will matter to you when you try to debug your program. Compile this program and run it. You may be surprised to find that it is the same program as the last one, except for the formatting. Don't get too worried about formatting style yet. You will have plenty of time to develop a style of your own as you learn the language. Be observant of styles as you see C programs in magazines, books, and other publications. This should pretty well cover the basic concepts of programming in C, but as there are many other things to learn, we will forge ahead to additional program structure.

PROGRAMMING EXERCISES

1. Write a program to display your name on the monitor.
2. Modify the program to display your address and phone number on separate lines by adding two additional "printf" statements.

oooooooooooOOOOOOOOOOOoooooooooo

FREE - A bug-free version

written by Mark Griffith et al
Part 2

[ED: This source code sample has come to us via the BitNet message system, from Mark Griffith.]

```
vol40   leax  -1,x
        beq   vol45

        exg   d,y
        addd  overflow,u
        exg   d,y
        addd  ,s
        bcc   vol40
        leay  1,y
        bra   vol40

* BRI:  clean up free clusters LSBs and old bytes in sector loop counter
vol45   leas  3,s        cleanup
*vol45  leas  2,s        cleanup

        std   nfree+2,u  Free sectors, low order
        sty   nfree,u    Free sectors, high order

*  Print volume status info  *

prnvol  lbsr   crlf
        leax  msg0,pcr   Point to "Volume; "
        bsr   pmsg       and print it
        lda   #$20       append a space
        bsr   listch     print that
        lda   #'"        Now a quote
        bsr   listch     print that
        leax  volname,u  Point to disk name
        bsr   pmsg       and print that
        lda   #'"        end with a quote
        bsr   listch     print that
        bsr   crlf       go to new line
        bsr   crlf       and another
        leax  msg3,pcr   Point to " Total:  "
        lbsr  pmsg       print it
        leax  total,u    Point to Total Sectors
        bsr   dblprez    Get the ASCII number
        leax  msg1,pcr   Point to " sectors ("
        lbsr  pmsg       print it
        leax  total,u    Point to bytes total
        ldd   1,x        multiply by 256
        std   ,x
        lda   3,x
        sta   2,x
        clr   3,x
        bsr   dblprez    Get the ASCII number
        leax  msg2,pcr   Point to " bytes)"
        bsr   pmsg       and print
        bsr   crlf       Print a newline
        leax  msg4,pcr   Point to " Free:   "
        lbsr  pmsg       print it
```

```
         leax  nfree,u    Point to sectors free
         bsr   dblprez    Get the ASCII number
         leax  msg1,pcr   Point to " sectors ("
         bsr   pmsg       print it
         leax  nfree,u    Point to bytes free
         ldd   1,x        multiply by 256
         std   ,x
         lda   3,x
         sta   2,x
         clr   3,x
         bsr   dblprez    Get the ASCII number
         leax  msg2,pcr   Point to " bytes)"
         bsr   pmsg       and print that too
         bsr   crlf       Do a couple newlines
         clrb             No errors
         lbra  exit       Finish up

         pag
*********************************
*
*       Subroutines
*

*  Print strings  *

pmsg     pshs  a,x
pmsg2    lda   ,x+
         beq   pmsg9
         bsr   listch
         tst   -1,x
         bpl   pmsg2
pmsg9    puls  a,x,pc

*  Send character in reg A to output  *

listch   pshs  d,x,y
         tfr   a,b
         bra   sendc1

crlf     pshs  d,x,y
         ldb   #$0d
         bra   sendc1

sendc    pshs  d,x,y
sendc1   andb  #$7f
         pshs  b
         tfr   s,x
         ldy   #1
         lda   #1
         os9   i$writln
         leas  1,s
sendc9   puls  d,x,y,pc

         pag
*********************************************
* Double precision (32 bit) binary to ASCII
*  In: (X) -> n  of 32 bits
```

```
*

hi16    equ  0
lo16    equ  2
digval  equ  4
dbliter equ  5
dblzflg equ  6
dblzch  equ  7
frames  set  dblzch+1

dblprez pshs d,x,y
        clra
        bra  dbl00
dblpre  PSHS D,X,Y
        lda  #$20
dbl00   leas -frames,s
        sta  dblzch,S
        ldd  hi16,x
        std  hi16,s
        ldd  lo16,x
        std  lo16,s

        lda  #dblitk    # powers within table
        sta  dbliter,s  loop iteration count
        leax dbltbl,pcr highest 10^n
        clr  dblzflg,s

dbl10   lda  #'0
        sta  digval,s

dbl20   ldd  lo16,S     low 16 of N
        subd lo16,X     minus 10^i
        tfr  d,y
        ldd  hi16,s
        sbcb hi16+1,X
        sbca hi16,X
        bcs  dbl30      subtract successful?
        sty  lo16,s     yes, save hi
        std  hi16,s     and low
        inc  digval,s
        bra  dbl20

dbl30   dec  dbliter,s  iteration count
        beq  dbl31      if last, must send a digit
        lda  digval,s   update leading 0 suppression
        suba #'0
        ora  dblzflg,s
        sta  dblzflg,s
        bne  dbl31
        lda  dblzch,s   get leading 0 suppressor
        beq  dbl32      if none
        sta  digval,s   space

dbl31   pshs x          save tbl pointer
        leax digval+2,s space or digit
        ldy  #1
        lda  #1
        os9   i$write
        puls  x
dbl32   leax  4,x        next
        tst   dbliter,S  maybe all done
        bne   dbl10      next power of ten

dbl80   leas  frames,s
        puls  d,x,y,pc

dbltbl
        fdb   $3B9A,$CA00 one billion

        fdb   $05F5,$E100 hundred million
        fdb   $0098,$9680 ten million
        fdb   $000F,$4240 one million

        fdb   $0001,$86A0 hundred thousand
        fdb   $0000,$2710 ten thousand
        fdb   $0000,$03E8 one thousand

        fdb   $0000,$0064 one hundred
        fdb   $0000,$000A ten
        fdb   $0000,$0001 one
dblitk  equ   (*-dbltbl)/4

*  Static Strings  *

msg0    fcs   "Volume: "
msg1    fcs   " sectors ("
msg2    fcs   " bytes) "
msg3    fcs   " Total:  "
msg4    fcs   " Free:   "

        emod
endmod  equ   *
        end
```

oooooooooo0000000000oooooooooo

OS9 Newsletter CONTENTS April '90 to June '91

Vol 4. No. 5 (Jun 90)

Page 02 Editorial	...Gordon Bentzen
Page 03 Creating a Boot disk	...Rob Unsworth
Page 05 CoCo3 Boot list order bug (BLOB)	...Kevin Darling
Page 08 OS9 from the beginning Pt 2	...Gordon Bentzen

Vol 4. No. 6 (Jul 90)

Page 02 Editorial	...Gordon Bentzen
Page 03 CC3Go and Shellplus	...Don Berrie
Page 04 How come my boot process takes so long	...Kevin Darling
Page 05 Building a Boot disk	...Rob Unsworth
Page 06 Check Book Programme	...Dale L.Puckett
Page 08 Window Directory	...Kevin Darling

Vol 4. No. 7 (Aug 90)

Page 02 Editorial	...Gordon Bentzen
Page 03 Shellscripts and the Menu command	...Bob Devries
Page 04 Introducing the MM/1	...Compuserve conference
Page 10 Installing a fan in your CoCo3	...From InterNet
Page 11 OS9 Magazines	...Peter Tutelaers

Vol 4. No. 8 (Sep 90)

Page 02 Editorial	...Gordon Bentzen
Page 03 Wanted: programme testers	...Don Berrie
Page 04 Computer Widow	...Val Bentzen
Page 04 Disk File fragmentation	...Bob Devries
Page 09 Dynacalc term string offsets	...Paul Good
Page 11 Modification to printer drivers	...Bob Devries

Vol 4. No. 9 (Oct 90)

Page 02 Editorial	...Gordon Bentzen
Page 03 PhantomGraph printer driver changes	...Bob Devries
Page 03 Basic09 Biomorphs	...Ted Martin
Page 07 OS9 Tutorial	...Bob Montowski
Page 09 Dynacalc Patch	...Don Berrie

Vol 4. No. 10 (Nov 90)

Page 02 Editorial	...Gordon Bentzen
Page 03 Which Computer?	...John Tynes
Page 04 OS9 Tutorial	...Bob Montowski
Page 07 Computer Languages	...David Solomon
Page 08 PD programme lists	

Vol 4. No. 11 (Dec 90)

oooooooooo0000000000oooooooooo

NATIONAL OS9 USER GROUP MEMBERS   as at 06/30/91 mm/dd/yy

AMBROSI	G. A.	172 Ogilvie Street	ESSENDON	Vic 3040	
BARBELER	Keith	37 Kunde Street	LOGANHOLME	Qld 4129	
BARKER	Bob	PO Box 711	LIVERPOOL	NSW 2170	
BENTZEN	Gordon	8 Odin Street	SUNNYBANK	Qld 4109	
BERRIE	Don	25 Irwin Terrace	OXLEY	Qld 4075	
BESASPARIS	Bernard	60 Power Street	NORMAN PARK	Qld 4170	
BISSELING	Fred	PO Box 770	COOMA	NSW 2630	
BROWN	Rohan	75 Pembroke Road	MOOROOLBARK	Vic 3138	
CALE	David	23 Hornsey Road	FLOREAT PARK	WA 6314	
CHARLESTON	Bob	7 Gaskin Avenue	HASTINGS	Vic 3915	
CHASE	Scott	3 Thomas Street	BAXTER	Vic 3911	
COOPER	L.A.	223 Elswick Street	LEICHARDT	NSW 2040	
COSSAR	Lin	12 Rakeiora Grove	PETONE	6303	NEW ZEALAND
DALZELL	Robbie	31 Nedland Crescent	PT.NOARLUNGA STH	SA 5167	
DEVRIES	Bob	21 Virgo Street	INALA	Qld 4077	
DONALDSON	Andrew	5 The Glades	DONCASTER	Vic 3108	
DONGES	Geoff	PO Box 326	KIPPAX	ACT 2615	
EASTHAM	David	87 Lewis Street	MARYVILLE	NSW 2293	
EATON	David	20 Gregson Place	CURTIN	ACT 2605	
EDWARDS	Peter	40 Davison Street	MITCHAM	Vic 3132	
EISEMAN	Fred	POBox228 freepost 12	CANNON HILL	Qld 4170	
ESKILDSEN	Ole	PO Box 496	PORT MORESBY		PAPUA NEW GUINEA
EVANS	John	80 Osburn Drive	MACGREGOR	ACT 2615	
FRANCIS	W.George	31 Donald Street	MORWELL	Vic 3840	
FRITZ	Terry	M/S 546	FOREST HILL	Qld 4342	
HARBECKE	Peter	18 Machenzie Street	MANLY WEST	Qld 4179	
HARRIS	Michael	PO Box 25	BELMORE	NSW 2192	
HARRY	Peter	13/51 Union Street	WINDSOR	Vic 3181	
HARTMANN	Alex	PO Box 1874	SOUTHPORT	Qld 4215	
HOLDEN	Rod	53 Haig Road	LOGANLEA	Qld 4131	
HUGHES	Peter	54 Princeton Street	KENMORE	Qld 4069	
HUTCHINSON	Simon	10 Ascot Court	NTH DANDENONG	Vic 3175	
IKIN	John	15 Seston Street	RESERVOIR	Vic 3073	
JACQUET	J.P.	27 Hampton Street	DURACK	Qld 4077	
KENDRICK	Gary	49 Morton Street	WEETANGERA	ACT 2614	
KINZEL	Burghard	Leipziger Ring 22A	5042 ERFTSTADT		GERMANY
MACKAY	Rob	7 Harburg drive	BEENLEIGH	Qld 4207	
MacKENZIE	Greg	346 Cook Road	HMAS CERBERUS	Vic 3920	WESTERN PORT
MARENTES	Nickolas	61 Cremin Street	UPPER MT.GRAVATT	Qld 4122	
MARTIN	Ted	PO Box 56	ROSNY PARK	Tas 7018	
McGIVERN	Jim	39 Bank Street	MEADOWBANK	NSW 2114	
McGRATH	A. John	93 Lemon Gums Drive	TAMWORTH	NSW 2340	
McLINTOCK	George	7 Logan Street	NARRABUNDAH	ACT 2604	
McMASTER	Brad	PO Box 1190	CROWS NEST	NSW 2065	
MORTON	David	c/o PO Box 195	CONDOBOLIN	NSW 2877	
MUNRO	Ron	45 Sunnyside Cres.	NORTH RICHMOND	NSW 2754	
PALMER	Brian	Unit 9/2 Para Street	BALGOWNIE	NSW 2519	
PATRICK	Wayne	12 O'Connell Street	GYMPIE	Qld 4570	
PEARCE	W.Leigh	47 Allenby Avenue	RESERVOIR	Vic 3073	
PENFOLD	Stephen	PO Box 385	MAITLAND	NSW 2320	
PETERSEN	Barry	4 Dargo Place	ALGESTER	Qld 4115	
PETERSEN	Mark	2 Shepherdson Street	CAPALABA	Qld 4157	
PRIMAVERA	Camillo	29 Richard Street	MARYBOROUGH	Qld 4650	
SINGER	Maurice	217 Preston Road	WYNNUM WEST	Qld 4178	
SKEBE	Jeff	23 Norma Road	PALM BEACH	NSW 2108	
STEPHEN	Val	1 Mabel Street	CAMBERWELL	Vic 3124	
SWINSCOE	Robin	17 Melaleuca Street	SLADE POINT	Qld 4740	
TARVIN	Digby	PO Box 498	RANDWICK	NSW 2031	
UNSWORTH	Rob	20 Salisbury Road	IPSWICH	Qld 4305	
VARNITI	Ken	2 Desindo Avenue	EDEN HILLS	SA 5050	
[illegible]	Paddie	Ercildoune Road	BURRUMBEET	Vic 3352	[illegible]
[illegible]	Mike	[illegible] Cir #174	LAS VEGAS		
[illegible]	Gord	51 Elizabeth Parade	LANE COVE	[illegible]	

AUSTRALIAN OS9 USER GROUP APPLICATION FORM
****

Surname : ________________________ First Name :______________________ Title (Mr.,Dr.,etc) : ____

Street : ______________________________

City / Suburb : ________________ State :___ Postcode : ____

Home Phone :_________________ Business Phone : _________________

Age Group (please tick) Under 18 [__] 18-25 [__] 26-35 [__] 36-45 [__] 46-55 [__] over 55 [__]

Do you run OS9 Level 1 [_] OS9 Level 2 [_] OSK [_] (please tick)

Type of Computer for OS9 :______________________________Memory Size :______________________

Diskette Size (inches): __ Number of Cylinders (tracks per side): __ Sides:__ Number of Drives :__

Printer Type: ____

Modem Type : ____

Other hardware : ____

Special Interests : ____

Can you contribute articles to this Newsletter :________

Date : ___/___/___ Signature :__________________

Amount Enclosed: $ _____ (**$18-00** will cover you for **12 months**)
(A$25-00 for overseas subscriptions)

Cheques made payable to NATIONAL OS9 USERGROUP please.

All subscriptions start with the September issue, and back issues will be sent where applicable.

Please Return Completed Form to :-

NATIONAL OS9 USER GROUP
C/o Gordon BENTZEN
8 ODIN STREET,
SUNNYBANK QLD 4109 (Phone 07 344 3881)
AUSTRALIA